WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Saturday, June 12, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ	
☐	L13	L12 and parallel	15
☐	L12	L9 and optimiz$	21
☐	L11	L10 and l5	2
☐	L10	L9 and VLIW	7
☐	L9	L7 and l3	28
		DB=EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L8	L7	1
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L7	L6 and (fetch$ or prefetch$)	50
☐	L6	L4 and interpret$	129
☐	L5	dynamic binary translat$	32
☐	L4	binary translat$	351
		DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ	
☐	L3	L2 or l1	7689
☐	L2	712/10-27,200-215,233-240.ccls.	5330
☐	L1	717/134-161.ccls.	2531

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 15 of 15 returned.

☐ 1. Document ID: US 20040088691 A1

Using default format because multiple data bases are involved.

L13: Entry 1 of 15 File: PGPB May 6, 2004

PGPUB-DOCUMENT-NUMBER: 20040088691
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040088691 A1

TITLE: Debugging and performance profiling using control-dataflow graph representations with reconfigurable hardware emulation

PUBLICATION-DATE: May 6, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Hammes, Jeffrey	Colorado Springs	CO	US	
Poznanovic, Daniel	Colorado Springs	CO	US	
Gliem, Lonnie	Burnsville	MN	US	

US-CL-CURRENT: 717/158; 717/134, 717/135

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | In

☐ 2. Document ID: US 20040088689 A1

L13: Entry 2 of 15 File: PGPB May 6, 2004

PGPUB-DOCUMENT-NUMBER: 20040088689
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040088689 A1

TITLE: System and method for converting control flow graph representations to control-dataflow graph representations

PUBLICATION-DATE: May 6, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Hammes, Jeffrey	Colorado Springs	CO	US	

US-CL-CURRENT: 717/154; 717/140, 717/155

ABSTRACT:

An embodiment of the invention includes a method of forming a control-dataflow graph that includes separating a control flow graph into two or more basic blocks, and converting said two or more basic blocks into code blocks, where the code blocks are formed into the control-dataflow graph. Another embodiment of the invention includes a method of forming a control-dataflow graph that includes separating a control flow graph into two or more basic blocks, forming a lode node in at least one of said basic blocks, forming a store node in at least one of said code blocks, inserting a delay node in at least one of said code blocks, segregating external hardware logic modules from said control flow graph, and converting said two or more basic blocks into code blocks, wherein the code blocks are formed into the control-dataflow graph.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | In

---

☐ 3. Document ID: US 20040088685 A1

L13: Entry 3 of 15 File: PGPB May 6, 2004

PGPUB-DOCUMENT-NUMBER: 20040088685
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040088685 A1

TITLE: Process for converting programs in high-level programming languages to a unified executable for hybrid computing platforms

PUBLICATION-DATE: May 6, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Poznanovic, Daniel	Colorado Springs	CO	US	
Hammes, Jeffrey	Colorado Springs	CO	US	
Krause, Lisa	Minneapolis	MN	US	
Steidel, Jon	Minneapolis	MN	US	
Barker, David	Salinas	CA	US	
Brooks, Jeffrey Paul	St. Louis	MN	US	

US-CL-CURRENT: 717/140; 717/107, 717/121

ABSTRACT:

A system and method for compiling computer code written to conform to a high-level language standard to generate a unified executable containing the hardware logic for a reconfigurable processor, the instructions for a traditional processor (instruction processor), and the associated support code for managing execution on a hybrid hardware platform. Explicit knowledge of writing hardware-level design code is not required since the problem can be represented in a high-level language syntax. A top-level driver invokes a standard-conforming compiler that provides syntactic and semantic analysis. The driver invokes a compilation phase that translates the CFG representation being generated into a hybrid controlflow-dataflow graph representation representing optimized pipelined logic which may be processed into a hardware description representation. The driver invokes a hardware description language (HDL) compiler to produce a netlist file that can be used to start the place-and-route compilation needed to produce a bitstream

for the reconfigurable computer. The programming environment then provides support for taking the output from the compilation driver and combining all the necessary components together to produce a unified executable capable of running on both the instruction processor and reconfigurable processor.

☐ 4. Document ID: US 20040044880 A1

L13: Entry 4 of 15 File: PGPB Mar 4, 2004

PGPUB-DOCUMENT-NUMBER: 20040044880
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040044880 A1

TITLE: Method and apparatus for transferring control in a computer system with dynamic compilation capability

PUBLICATION-DATE: March 4, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Altman, Erik R.	Danbury	CT	US	
Ebcioglu, Kemal	Katonah	NY	US	
Gschwind, Michael Karl	Mohegan Lake	NY	US	
Luick, David Arnold	Rochester	MN	US	

US-CL-CURRENT: 712/209; 712/227

ABSTRACT:

In a dynamically compiling computer system, a system and method for efficiently transferring control from execution of an instruction in a first representation to a second representation of the instruction is disclosed. The system and method include the setting of a tag for entry points of each instruction in a first representation that has been translated to a second representation. The tag is stored in memory in association with each such instruction. When a given instruction in a first representation is to be executed, the tag is examined, and if it indicates that a translated version of the instruction has previously been generated, control is passed to execution of the instruction in the second representation. The second representation can be a different instruction set representation, or an optimized representation in the same instruction set as the original instruction.

☐ 5. Document ID: US 20040015888 A1

L13: Entry 5 of 15 File: PGPB Jan 22, 2004

PGPUB-DOCUMENT-NUMBER: 20040015888

PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040015888 A1

TITLE: Processor system including dynamic translation facility, binary translation program that runs in computer having processor system implemented therein, and semiconductor device having processor system implemented therein

PUBLICATION-DATE: January 22, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Fujii, Hiroaki	Tokorozawa		JP	
Tanaka, Yoshikazu	Tokorozawa		JP	
Miki, Yoshio	Kodaira		JP	

US-CL-CURRENT: 717/136; 717/153

ABSTRACT:

An interpretation flow, a translation and optimization flow, and an original instruction prefetch flow are defined independently of one another. A processor is realized as a chip multiprocessor or realized so that one instruction execution control unit can process a plurality of processing flows simultaneously. The plurality of processing flows is processed in parallel with one another. Furthermore, within the translation and optimization flow, translated instructions are arranged to define a plurality of processing flows. Within the interpretation flow, when each instruction is interpreted, if a translated instruction corresponding to the instruction processed within the translation and optimization flow is present, the translated instruction is executed. According to the present invention, an overhead including translation and optimization that are performed in order to execute instructions oriented to an incompatible processor is minimized. At the same time, translated instructions are processed quickly, and a processor is operated at a high speed with low power consumption. Furthermore, an overhead of original instruction fetching is reduced.

☐ 6. Document ID: US 20030182653 A1

L13: Entry 6 of 15 File: PGPB Sep 25, 2003

PGPUB-DOCUMENT-NUMBER: 20030182653
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20030182653 A1

TITLE: Systems and methods for verifying correct execution of emulated code via dynamic state verification

PUBLICATION-DATE: September 25, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Desoli, Giuseppe	Watertown	MA	US	

Bala, Vasanth	Tarrytown	NY	US
Duesterwald, Evelyn	Somerville	MA	US

US-CL-CURRENT: 717/138; 703/23, 717/139, 719/328

ABSTRACT:

Systems and methods for verifying execution of translated code operative on a host computer system different from the computer system designated for the original program code. In one arrangement, the system and method fetch program code, translate program code, emit the translated program code into at least one code cache, execute the translated code within the at least one code cache, interpret the program code, and compare a translator generated state with an interpreter generated state to confirm desired code execution.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | In

---

☐ 7. Document ID: US 20030101439 A1

L13: Entry 7 of 15 File: PGPB May 29, 2003

PGPUB-DOCUMENT-NUMBER: 20030101439
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20030101439 A1

TITLE: System and method for supporting emulation of a computer system through dynamic code caching and transformation

PUBLICATION-DATE: May 29, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Desoli, Giuseppe	Watertown	MA	US	
Bala, Vasanth	Sudbury	MA	US	
Duesterwald, Evelyn	Somerville	MA	US	

US-CL-CURRENT: 717/148; 717/138, 717/139

ABSTRACT:

The present disclosure relates to a system and method for emulating a computer system. In one arrangement, the system and method pertain to fetching program code, translating program code, emitting translated program code into at least one code cache, and executing translated code within the at least one code cache in lieu of associated program code when a semantic function of the associated program code is requested. Operation of the system and method can be facilitated with an application programming interface that, in one arrangement, can comprise a set of functions available to the translator including an emit fragment function with which the translator can emit code fragments into code caches of the dynamic execution layer interface, and an execute function with which the translator can request execution of code fragments contained within the at least one code cache.

☐ 8. Document ID: US 20020199179 A1

L13: Entry 8 of 15 File: PGPB Dec. 26, 2002

PGPUB-DOCUMENT-NUMBER: 20020199179
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020199179 A1

TITLE: Method and apparatus for compiler-generated triggering of auxiliary codes

PUBLICATION-DATE: December 26, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Lavery, Daniel M.	Santa Clara	CA	US	
Wang, Hong	Fremont	CA	US	
Hoflehner, Gerolf F.	Santa Clara	CA	US	
Liao, Shih-wei	Sunnyvale	CA	US	
Shen, John	San Jose	CA	US	
Grochowski, Edward T.	San Jose	CA	US	
Sehr, David C.	Sunnyvale	CA	US	
Fang, Jesse Z.	San Jose	CA	US	

US-CL-CURRENT: 717/158

ABSTRACT:

A method for executing a code is provided. The method includes receiving a trigger instruction, selecting an entry in a trigger table, the entry associated with the trigger instruction, and executing an auxiliary code referenced by the entry in the trigger table.

☐ 9. Document ID: US 6691306 B1

L13: Entry 9 of 15 File: USPT Feb 10, 2004

US-PAT-NO: 6691306
DOCUMENT-IDENTIFIER: US 6691306 B1

TITLE: Use of limited program space of general purpose processor for unlimited sequence of translated instructions

DATE-ISSUED: February 10, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Cohen; Ariel	Cupertino	CA		
Perets; Ronen	Cupertino	CA		
Zemlyak; Boris	Cupertino	CA		

US-CL-CURRENT: 717/139; 703/26, 711/213, 712/208, 712/209, 717/136, 717/137, 717/140, 717/148

ABSTRACT:

An apparatus comprising a circuit configured to (i) translate one or more instruction codes of a first instruction set into a sequence of instruction codes of a second instruction set and (ii) present the sequence of instruction codes of the second instruction set in response to a predetermined number of addresses.

22 Claims, 12 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 7

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw. Desc | Ir

☐ 10. Document ID: US 6463582 B1

L13: Entry 10 of 15 File: USPT Oct 8, 2002

US-PAT-NO: 6463582
DOCUMENT-IDENTIFIER: US 6463582 B1

TITLE: Dynamic optimizing object code translator for architecture emulation and dynamic optimizing object code translation method

DATE-ISSUED: October 8, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Lethin; Richard A.	New York	NY		
Bank, III; Joseph A.	New York	NY		
Garrett; Charles D.	Seattle	WA		
Wada; Mikayo	Kawasaki			JP
Sakurai; Mitsuo	Kawasaki			JP

US-CL-CURRENT: 717/158; 717/138, 717/139

ABSTRACT:

An optimizing object code translation system and method perform dynamic compilation and translation of a target object code on a source operating system while performing optimization. Compilation and optimization of the target code is dynamically executed in real time. A compiler performs analysis and optimizations that improve emulation relative to template-based translation and interpretation such that a host processor which processes larger order instructions, such as 32-bit instructions, may emulate a target processor which processes smaller order instructions, such as 16-bit and 8-bit instructions. The optimizing object code translator does not require knowledge of a static program flow graph or memory locations of target instructions prior to run time. In addition, the optimizing object code translator does not require knowledge of the location of all join points into the target object code prior to execution. During program execution, a translator records branch operations. The logging of information identifies instructions and instruction join points. When a number of times a branch operation is executed exceeds a threshold, the destination of the branch becomes a seed for compilation and code portions between seeds are defined as segments. A segment may be incomplete allowing for modification or replacement to account for a new flow of program control during real time program execution.

32 Claims, 37 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 19

☐ 11. Document ID: US 6397379 B1

L13: Entry 11 of 15 File: USPT May 28, 2002

US-PAT-NO: 6397379
DOCUMENT-IDENTIFIER: US 6397379 B1

TITLE: Recording in a program execution profile references to a memory-mapped active device

DATE-ISSUED: May 28, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Yates, Jr.; John S.	Needham	MA		
Reese; David L.	Westborough	MA		
Van Dyke; Korbin S.	Sunol	CA		

US-CL-CURRENT: 717/140

ABSTRACT:

A method and a computer for execution of the method. As part of executing a stream of instructions, a series of memory loads is issued from a computer CPU to a bus, some directed to well-behaved memory and some directed to non-well-behaved devices in I/O space. Computer addresses are stored of instructions of the stream that issued memory loads to the non-well-behaved memory, the storage form of the recording allowing determination of whether the memory load was to well-behaved memory or not-well-behaved memory without resolution of any memory address stored in the recording.

47 Claims, 5 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 41

☐ 12. Document ID: US 6311261 B1

L13: Entry 12 of 15 File: USPT Oct 30, 2001

US-PAT-NO: 6311261
DOCUMENT-IDENTIFIER: US 6311261 B1

TITLE: Apparatus and method for improving superscalar processors

DATE-ISSUED: October 30, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chamdani; Joseph I.	Marietta	GA		
Alford; Cecil O.	Lawrenceville	GA		

US-CL-CURRENT: 712/23

ABSTRACT:

The invention involves new microarchitecture apparatus and methods for superscalar microprocessors that support multi-instruction issue, decoupled dataflow scheduling, out-of-order execution, register renaming, multi-level speculative execution, and precise interrupts. These are the Distributed Instruction Queue (DIQ) and the Modified Reorder Buffer (MRB). The DIQ is a new distributed instruction shelving technique that is an alternative to the reservation station (RS) technique and offers a more efficient (improved performance/cost) implementation. The Modified Reorder Buffer (MRB) is an improved reorder buffer (RB) result shelving technique eliminates the slow and expensive prioritized associative lookup, shared global buses, and dummy branch entries (to reduce entry usage). The MRB has an associateive key unit which uses a unique associative key.

14 Claims, 40 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 40

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Ir

☐ 13. Document ID: US 6223339 B1

L13: Entry 13 of 15 File: USPT Apr 24, 2001

US-PAT-NO: 6223339
DOCUMENT-IDENTIFIER: US 6223339 B1
**** See image for Certificate of Correction ****

TITLE: System, method, and product for memory management in a dynamic translator

DATE-ISSUED: April 24, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY

Shah; Lacky V.	Sunnyvale	CA
Mattson, Jr.; James S.	Campbell	CA
Buzbee; William B.	Half Moon Bay	CA

US-CL-CURRENT: 717/158; 717/153

ABSTRACT:

The present invention is a system, method, and product for improving the speed of dynamic translation systems by efficiently positioning translated instructions in a computer memory unit. More specifically, the speed of execution of translated instructions, which is a factor of particular relevance to dynamic optimization systems, may be adversely affected by inefficient jumping between traces of translated instructions. The present invention efficiently positions the traces with respect to each other and with respect to "trampoline" instructions that redirect control flow from the traces. For example, trampoline instructions may redirect control flow to an instruction emulator if the target instruction has not been translated, or to the translation of a target instruction that has been translated. When a target instruction has been translated, a backpatcher of the invention may directly backpatch the jump to the target so that the trampoline instructions are no longer needed. A method of the present invention includes: (1) designating "chunks" of memory locations, and (2) positioning a translated trace and its corresponding trampoline instructions in the same chunk. The size of the chunk generally is based on a "machine-specific shortest jump distance" that is the shortest maximum distance that a jump instruction may specify. In particular, the chunk length may be determined so that, for every translated trace and trampoline instruction positioned in the same chunk, the greatest distance between a translated jump instruction and its target trampoline instruction is not greater than the machine-specific shortest jump distance for that type of jump instruction.

72 Claims, 6 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 6

☐ 14. Document ID: US 6189141 B1

L13: Entry 14 of 15 File: USPT Feb 13, 2001

US-PAT-NO: 6189141
DOCUMENT-IDENTIFIER: US 6189141 B1

TITLE: Control path evaluating trace designator with dynamically adjustable thresholds for activation of tracing for high (hot) activity and low (cold) activity of flow control

DATE-ISSUED: February 13, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Benitez; Manuel E.	Cupertino	CA		
Mattson, Jr.; James S.	Campbell	CA		
Buzbee; William B.	Half Moon Bay	CA		

Shah; Lacky V.	Sunnyvale	CA

US-CL-CURRENT: 717/153; 717/156, 717/158

ABSTRACT:

A computer-implemented system, method, and product are provided to designate and translate traces of original instructions of an executable file at run time based on dynamic evaluation of control flow through frequently executed traces of instructions. Such designation typically reduces unnecessary translations and optimizations, and thereby increases execution speed and reduces the usage of memory and other resources. The invention includes a hot trace identifier to identify frequently executed traces of instructions and a hot trace instrumenter to instrument such frequently executed traces so that control flow through them may be recorded. If the amount or rate of control flow through a frequently executed trace exceeds a threshold value, a hot trace selector is invoked to select a hot trace of original instructions including those of the frequently executed trace. The hot trace may be dynamically optimized. The system, method, and product also provide for the continuous recording of control flow through hot traces. If control flow has changed during execution, such that the amount or rate of control flow through a hot trace falls below a threshold value, the trace may be removed.

69 Claims, 15 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 13

☐ 15. Document ID: US 6112019 A

L13: Entry 15 of 15 File: USPT Aug 29, 2000

US-PAT-NO: 6112019
DOCUMENT-IDENTIFIER: US 6112019 A
**** See image for Certificate of Correction ****

TITLE: Distributed instruction queue

DATE-ISSUED: August 29, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chamdani; Joseph I.	Marietta	GA		
Alford; Cecil O.	Lawrenceville	GA		

US-CL-CURRENT: 712/214

ABSTRACT:

A distributed instruction queue (DIQ) in a superscalar microprocessor supports multi-instruction issue, decoupled data flow scheduling, out-of-order execution, register renaming, multi-level speculative execution, and precise interrupts. The DIQ provides distributed instruction shelving without storing register values, operand value copying, and result value forwarding, and supports in-order issue as well as out-of-order issue

within its functional unit. The DIQ allows a reduction in the number of global wires and replacement with private-local wires in the processor. The DIQ's number of global wires remains the same as the number of DIQ entries and data size increases. The DIQ maintains maximum machine parallelism and the actual performance of the microprocessor using the DIQ is better due to reduced cycle time or more operations executed per cycle.

22 Claims, 40 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 40

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw Desc | In

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Terms	Documents
L12 and parallel	15

Display Format: - Change Format

Previous Page   Next Page   Go to Doc#

YAHOO! search dynamic binary translation optimization Yahoo! Search Advance Preferen

Web Images Directory Yellow Pages News Products

TOP 20 WEB RESULTS out of about 63,100 Search took 0.196 seconds. (What's this?)

1. **Dynamic Binary Translation** and **Optimization**
   Abstract—We describe a VLIW architecture designed specifically as a target for **dynamic** compilation of an existin just-in-time compilation, adaptive code ...
   www.computer.org/tc/tc2001/t0529abs.htm - 11k - Cached - More pages from this site

2. **Dynamic** and Transparent **Binary Translation** - Gschwind, Altman, Sathaye, Ledak, Appenze
   We describe an implementation of the PowerPC architecture using **dynamic** compilation techniques to an optimiz - **Dynamic Binary**.. (Correct) **Dynamic Optimization** of Micro-Operations - Brian Slechta David (2003) ...
   citeseer.nj.nec.com/423478.html - 20k - Cached - More pages from this site

3. **Dynamic** and Transparent **Binary Translation**
   Dynamic and Transparent **Binary Translation** We describe an implementation of the PowerPC architecture usin
   citeseer.ist.psu.edu/423478.html - More pages from this site

4. Micro-33 December 13, 2000 (PDF)
   ... **Dynamic Binary Translation** and **Optimization**Erik R. AltmanKemal Ebcioˇglu IBM T.J ... for Micro-33 Tutorial
   www.microarch.org/micro33/tutorial/m33-t-issues.pdf - 79k - View as html - More pages from this site

5. **Dynamic** and Transparent **Binary Translation**
   ... microprocessor implementations. The **Binary-translation** Optimized Architecture (BOA), an ... PowerPC family
   www.computer.org/computer/co2000/r3054abs.htm - 10k - Cached - More pages from this site

6. Transparent **Dynamic Optimization**: The Design and Implementation of Dynamo (PDF)
   ... Transparent **Dynamic Optimization**: The Design and Implementation of DynamoVasanth ... hp.com**dynamicc**
   www.hpl.hp.com/techreports/1999/HPL-1999-78.pdf - 617k - View as html - More pages from this site

7. Security Applications of **Dynamic Binary Translation** (PDF)
   ... SECURITY APPLICATIONS OF **DYNAMIC BINARY TRANSLATION**byDINO DAI ZOVI ... application of **dyna**
   www.sandia.gov/iorta/docs/ddz_thesis.pdf - 122k - View as html

8. Cristina Cifuentes: Publications in the Area of **Binary Translation**
   ... A Retargetable **Dynamic Binary Translation** Framework, Fourth ... **Dynamic Binary Translation**. Proceeding
   www.sunlabs.com/people/cristina/binarytranslation-publications.html - 15k - Cached - More pages from this site

9. SIND: A Framework for **Binary Translation** (PDF)
   ... techniques of **binary translation**. Current research focuses are **dynamic optimization**. of running ... by everyo
   www.dyregod.dk/uni/artikler/sind-a-framework-for.pdf - 68k - View as html

10. SECURITY APPLICATIONS OF **DYNAMIC BINARY TRANSLATION** by DINO DAI ZOVI THE
    ... SECURITY APPLICATIONS OF **DYNAMIC BINARY TRANSLATION**byDINO DAI ZOVI ... application of **dyna**
    www.cs.unm.edu/~ghandi/ddz-thesis.pdf - 115k - View as html - More pages from this site

11. CS851 001: Topics in Software **Dynamic Translation**
    ... Feedback-Directed and **Dynamic Optimization**. **Dynamic Binary Translation** and **Optimization** by Kemal Eb
    www.cs.virginia.edu/~jks6b/cs851 - 7k - Cached - More pages from this site

12. Michael Gschwind
... Albonesi, S. Dwarkadas) **Dynamic Binary Translation** and **Optimization**. IEEE Transactions on Computers ..
www.research.ibm.com/people/m/mikeg - 28k - Cached - More pages from this site

13. VLIW at IBM Research
... software **dynamic binary translation** ... **Binary translation** eliminates the **binary** compatibility problem faced
www.research.ibm.com/vliw/abs.html - 49k - Cached - More pages from this site

14. Walkabout-A Retargetable **Dynamic Binary Translation** Framework
... develops **dynamic binary translation** techniques ... **dynamic binary translation** ideas, as well as techniques
www.sunlabs.com/techrep/2002/abstract-106.html - 11k - Cached - More pages from this site

15. On the Code Quality of **Dynamic Binary** Translators (PDF)
Bi-nary **translation**, on the other hand, in which the **binary** code for one machineis recompiled for another, can le
translation, in whichthe **translation** is performed on ... We show that **dynamic binary translation** is one**dynam**
savage.light-speed.de/pdf/cqual-dbt.pdf - 183k - View as html

16. Tech Report: HPL-1999-78: Transparent **Dynamic Optimization**: The
... **dynamic optimization**; compiler; trace selection; **binary translation**. Abstract: **Dynamic optimization** refers t
www.hpl.hp.com/techreports/1999/HPL-1999-78.html - 5k - Cached - More pages from this site

17. Dino Dai Zovi
... application of **dynamic binary translation**, previously a technique primarily for **dynamic optimization**, to stop
www.cs.unm.edu/~ghandi - 6k - Cached - More pages from this site

18. Retargetable and Reconfigurable Software **Dynamic Translation** (PDF)
... Retargetable and Reconfigurable Software **Dynamic Translation**K. Scott*, N ... the broader literature in**dynan**
www.cs.pitt.edu/~naveen/papers/cgo-03.pdf - 235k - View as html - More pages from this site

19. 3rd Annual Workshop on Computer Architecture Education
... Commercializing **Binary Translation**. **Binary translation** and **optimization** have reached a ... is commercializ
www.ece.neu.edu/info/architecture/a.html - 5k - Cached

20. Retargetable and Reconfigurable Software **Dynamic Translation** (PDF)
... AbstractSoftware **dynamic translation** (SDT) is a technologythat permits the ... broader literature in **dynamic o**
www.cs.virginia.edu/~jks6b/papers/cgo03.pdf - 157k - View as html - More pages from this site

Results Page:
1 2 3 4 5 6 7 8 9 10 ▶ **Next**

Help us improve your search experience. Send us feedback.

Web | **Images** | **Directory** | **Yellow Pages** | **News** | **Products**

Your Search: dynamic binary translation optimization [Yahoo! Search] Advanced Web Search Preferences

Yahoo! Search is hiring! Learn about job opportunities

Search with your friends with the Yahoo! Search IMVironment

Google

Web Images Groups News Froogle **more »**

dynamic "binary translation" optimization | Search | Advanced Search Preferences

Web Results **1** - **10** of about **1,860** for dynamic "binary translation" optimization. (**0.21** seconds)

Dynamic Binary Translation and Optimization
MICRO-33 Presents a Tutorial: **Dynamic Binary Translation** and **Optimization** Wednesday, December 13, 2000 2:30 - 6:00 pm Monterey Plaza Hotel Monterey, California. ...
www.microarch.org/micro33/tutorial/tutorial.html - 6k - Cached - Similar pages

[PDF] **Dynamic Binary Translation** and **Optimization** Erik R. Altman Kemal ...
File Format: PDF/Adobe Acrobat - View as HTML
Page 1. **Dynamic Binary Translation** and **Optimization** Erik R. Altman Kemal Ebcio˘glu IBM TJ Watson Research Center Micro-33 December 13, 2000 Page 2. ...
www.microarch.org/micro33/tutorial/m33-t-issues.pdf - Similar pages
[More results from www.microarch.org]

Dynamic Binary Translation and Optimization
Technical Paper Search Technical Paper. **Dynamic Binary Translation** and **Optimization** We describe a VLIW architecture designed specifically ...
domino.watson.ibm.com/library/cyberdig.nsf/ 0/4039c65f62ee3d0685256a2900529e89?OpenDocument - 13k - Cached - Similar pages

[PDF] **Dynamic Binary Translation** and **Optimization**
File Format: PDF/Adobe Acrobat - View as HTML
New Course Announcement for Spring 2001 **Dynamic Binary Translation** and **Optimization** Csci 8980 – Spring 2001 Wei Hsu TTh 12:45pm-2:00pm in room 208, 1701 Univ ...
www-users.cs.umn.edu/~hsu/courses/8980.pdf - Similar pages

Dynamic Binary Translation and Optimization
... **Dynamic Binary Translation** and **Optimization**. Full text, Full text available on the Publisher sitePublisher Site. Source, IEEE Transactions ...
portal.acm.org/citation.cfm?id=500509& dl=ACM&coll=portal&CFID=11111111&CFTOKEN=2222222 - Similar pages

Dynamic binary translation for accumulator-oriented architectures
... 14 Kemal Ebcioglu , Erik Altman , Michael Gschwind , Sumedh Sathaye, **Dynamic Binary Translation** and **Optimization**, IEEE Transactions on Computers, v.50 n.6, p ...
portal.acm.org/citation.cfm?id=776264& dl=ACM&coll=portal&CFID=11111111&CFTOKEN=2222222 - Similar pages
[More results from portal.acm.org]

Dynamic Binary Translation and Optimization
cs-ieee-logo. Search. Help. Contact. Shopping Cart. YOU ARE BEING REDIRECTED. PLEASE CORRECT YOUR BOOKMARK. **Dynamic Binary Translation** and **Optimization**.
www.computer.org/tc/tc2001/t0529abs.htm - 5k - Cached - Similar pages

[PDF] Microsoft PowerPoint - **Dynamic Binary Translation**
File Format: PDF/Adobe Acrobat - View as HTML
... et al., Dynamo: A Transparent **Dynamic Optimization** System, PLDI ... Special issue on **Dynamic** Optimisation IEEE ... March 2000 – Special issue on **Binary Translation**
lapwww.epfl.ch/courses/advcomparch/ Dynamic%20Binary%20Translation.pdf - Similar pages

[PDF] Walkabout – A Retargetable **Dynamic Binary Translation** Framework
File Format: PDF/Adobe Acrobat - View as HTML

Walkabout – A Retargetable **Dynamic Binary Translation** Framework Cristina Cifuentes, Brian Lewis and David Ung Sun Microsystems Laboratories Palo Alto, CA ...
www.itee.uq.edu.au/~cristina/wbt02.pdf - Similar pages

CS851 001: Topics in Software **Dynamic** Translation
... **Dynamic Binary Translation** and **Optimization** by Kemal Ebcioglu, Erik Altman, Michael Gschwind, and Sumed Sathaye. In IEEE Transactions on Computers. ...
www.cs.virginia.edu/~jks6b/cs851/ - 8k - Cached - Similar pages

PORTAL

US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: [✓ The ACM Digital Library] The Guide

+dynamic +binary-translation +interpreter +optimiz%

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **dynamic binary translation interpreter optimiz%**

Found **39** of **18,467 searched out of 137,188.**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 39

Result page: **1** 2 next

Relevance scale

1 Dynamic native optimization of interpreters

Gregory T. Sullivan, Derek L. Bruening, Iris Baron, Timothy Garnett, Saman Amarasinghe

June 2003 **Proceedings of the 2003 workshop on Interpreters, Virtual Machines and Emulators**

Full text available: pdf(150.25 KB) Additional Information: full citation, abstract, references, index terms

For domain specific languages, "scripting languages", dynamic languages, and for virtual machine-based languages, the most straightforward implementation strategy is to write an interpreter. A simple interpreter consists of a loop that fetches the next bytecode, dispatches to the routine handling that bytecode, then loops. There are many ways to improve upon this simple mechanism, but as long as the execution of the program is driven by a representation of the program other than as a stream of n ...

2 Dynamic translation: The Transmeta Code Morphing™ Software: using speculation, recovery, and adaptive retranslation to address real-life challenges

James C. Dehnert, Brian K. Grant, John P. Banning, Richard Johnson, Thomas Kistler, Alexander Klaiber, Jim Mattson

March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**

Full text available: pdf(988.25 KB) Publisher Site

Additional Information: full citation, abstract, references

Transmeta's Crusoe microprocessor is a full, system-level implementation of the x86 architecture, comprising a native VLIW microprocessor with a software layer, the **Code Morphing Software** (CMS), that combines an interpreter, dynamic binary translator, optimizer, and runtime system. In its general structure, CMS resembles other binary translation systems described in the literature, but it is unique in several respects. The wide range of PC workloads that CMS must handle gracefully in real ...

Keywords: binary translation, dynamic optimization, dynamic translation, emulation, self-modifying code, speculation

3 Machine-adaptable dynamic binary translation

David Ung, Cristina Cifuentes

January 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN workshop on Dynamic and adaptive compilation and optimization**, Volume 35 Issue 7

Full text available: pdf(1.23 MB) Additional Information: full citation, abstract, references, citings, index terms

Dynamic binary translation is the process of translating and optimizing executable code for one machine to another at runtime, while the program is "executing" on the target machine.

Dynamic translation techniques have normally been limited to two particular machines; a competitor's machine and the hardware manufacturer's machine. This research provides for a more general framework for dynamic translations, by providing a framework based on specifications of machines that ...

Keywords: binary translation, dynamic compilation, dynamic execution, emulation, interpretation

4 Hardware Support for Control Transfers in Code Caches
Ho-Seop Kim, James E. Smith
December 2003 **Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture**
Full text available: pdf(315.74 KB) Publisher Site Additional Information: full citation, abstract

Many dynamic optimization and/or binary translationsystems hold optimized/translated superblocks in a codecache. Conventional code caching systems suffer fromoverheads when control is transferred from one cachedsuperblock to another, especially via register-indirectjumps. The basic problem is that instruction addresses inthe code cache are different from those in the original programbinary. Therefore, performance for register-indirectjumps depends on the ability to translate efficiently fromsour ...

5 A brief history of just-in-time
John Aycock
June 2003 **ACM Computing Surveys (CSUR)**, Volume 35 Issue 2
Full text available: pdf(171.09 KB) Additional Information: full citation, abstract, references, index terms

Software systems have been using "just-in-time" compilation (JIT) techniques since the 1960s. Broadly, JIT compilation includes any translation performed dynamically, after a program has started execution. We examine the motivation behind JIT compilation and constraints imposed on JIT compilation systems, and present a classification scheme for such systems. This classification emerges as we survey forty years of JIT work, from 1960--2000.

Keywords: Just-in-time compilation, dynamic compilation

6 Optimization and precise exceptions in dynamic compilation
Michael Gschwind, Erik Altman
March 2001 **ACM SIGARCH Computer Architecture News**, Volume 29 Issue 1
Full text available: pdf(508.52 KB) Additional Information: full citation, abstract, index terms

Maintaining precise exceptions is an important aspect of achieving full compatibility with a legacy architecture. While asynchronous exceptions can be deferred to an appropriate boundary in the code, synchronous exceptions must be taken when they occur. This introduces uncertainty into liveness analysis since processor state that is otherwise dead may be exposed when an exception handler is invoked. Previous systems either had to sacrifice full compatibility to achieve more freedom to perform op ...

7 Dynamo: a transparent dynamic optimization system
Vasanth Bala, Evelyn Duesterwald, Sanjeev Banerjia
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation**, Volume 35 Issue 5
Full text available: pdf(156.03 KB) Additional Information: full citation, abstract, references, citings, index terms

We describe the design and implementation of Dynamo, a software dynamic optimization system that is capable of transparently improving the performance of a native instruction stream as it executes on the processor. The input native instruction stream to Dynamo can be dynamically generated (by a JIT for example), or it can come from the execution of a statically compiled native binary. This paper evaluates the Dynamo system in the latter, more challenging situation, in order to emphasize the ...

8 An out-of-order execution technique for runtime binary translators
Bich C. Le
October 1998 **Proceedings of the eighth international conference on Architectural support for programming languages and operating systems**, Volume 32 , 33 Issue 5 , 11
Full text available: pdf(1.04 MB) Additional Information: full citation, abstract, references, citings, index terms

A dynamic translator emulates an instruction set architccturc by translating source instructions to native code during execution. On statically-scheduled hardware, higher performance can potentially be achieved by reordering the translated instructions; however, this is a challenging transformation if the source architecture supports precise exception semantics, and the user-level program is allowed to register exception handlers. This paper presents a software technique which allows a translato ...

9 Optimizations and oracle parallelism with dynamic translation
Kemal Ebcioğlu, Erik R. Altman, Michael Gschwind, Sumedh Sathaye
November 1999 **Proceedings of the 32nd annual ACM/IEEE international symposium on Microarchitecture**
Full text available: pdf(1.28 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

We describe several optimizations which can be employed in a dynamic binary translation (DBT) system, where low compilation/translation overhead is essential. These optimizations achieve a high degree of ILP, sometimes even surpassing a static compiler employing more sophisticated, and more time-consuming algorithms [9]. We present results in which we employ these optimizations in a dynamic binary translation system capable of computing oracle parallelism.

10 Profile-based optimizations: Dynamic trace selection using performance monitoring hardware sampling
Howard Chen, Wei-Chung Hsu, Jiwei Lu, Pen-Chung Yew, Dong-Yuan Chen
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**
Full text available: pdf(1.88 MB) Publisher Site Additional Information: full citation, abstract, references

Optimizing programs at run-time provides opportunities to apply aggressive optimizations to programs based on information that was not available at compile time. At run time, programs can be adapted to better exploit architectural features, optimize the use of dynamic libraries, and simplify code based on run-time constants.Our profiling system provides a framework for collecting information required for performing run-time optimization. We sample the performance hardware registers available on ...

11 Binary translation and architecture convergence issues for IBM system/390
Michael Gschwind, Kemal Ebcioğlu, Erik Altman, Sumedh Sathaye
May 2000 **Proceedings of the 14th international conference on Supercomputing**

Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, references, index terms

We describe the design issues in an implementation of the ESA/390 architecture based on binary translation to a very long instruction word (VLIW) processor. During binary translation, complex ESA/390 instructions are decomposed into instruction "primitives" which are then scheduled onto a wide-issue machine. The aim is to achieve high instruction level parallelism due to the increased scheduling and optimization opportunities which can be exploited by binary translation software ...

12 A study of exception handling and its dynamic optimization in Java
Takeshi Ogasawara, Hideaki Komatsu, Toshio Nakatani
October 2001 **ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications**, Volume 36 Issue 11

Full text available: pdf(190.18 KB) Additional Information: full citation, abstract, references, citings, index terms

Optimizing exception handling is critical for programs that frequently throw exceptions. We observed that there are many such exception-intensive programs iin various categories of Java programs. There are two commonly used exception handling techniques, stack unwinding optimizes the normal path, while stack cutting optimizes the exception handling path. However, there has been no single exception handling technique to optimize both paths.

13 Dynamic Adaptive compilation: An infrastructure for adaptive dynamic optimization
Derek Bruening, Timothy Garnett, Saman Amarasinghe
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**

Full text available: pdf(1.16 MB) Publisher Site Additional Information: full citation, abstract, references, citings

Dynamic optimization is emerging as a promising approach to overcome many of the obstacles of traditional static compilation. But while there are a number of compiler infrastructures for developing static optimizations, there are very few for developing dynamic optimizations. We present a framework for implementing dynamic analyses and optimizations. We provide an interface for building external modules, or clients, for the DynamoRIO dynamic code modification system. This interface abstracts awa ...

14 Practicing JUDO: Java under dynamic optimizations
Michał Cierniak, Guei-Yuan Lueh, James M. Stichnoth
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation**, Volume 35 Issue 5

Full text available: pdf(190.06 KB) Additional Information: full citation, abstract, references, citings, index terms

A high-performance implementation of a Java Virtual Machine (JVM) consists of efficient implementation of Just-In-Time (JIT) compilation, exception handling, synchronization mechanism, and garbage collection (GC). These components are tightly coupled to achieve high performance. In this paper, we present some static anddynamic techniques implemented in the JIT compilation and exception handling of the Microprocessor Research Lab Virtual Machine (MRL VM), ...

15 Novel ideas: Performance characterization of a hardware mechanism for dynamic optimization
Brian Fahs, Satarupa Bose, Matthew Crum, Brian Slechta, Francesco Spadini, Tony Tung, Sanjay J. Patel, Steven S. Lumetta
December 2001 **Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture**
Full text available: pdf(1.31 MB) Additional Information: full citation, abstract, references, citings
Publisher Site

We evaluate the rePLay microarchitecture as a means for reducing application execution time by facilitating dynamic optimization. The framework contains a programmable optimization engine coupled with a hardware-based recovery mechanism. The optimization engine enables the dynamic optimizer to run concurrently with program execution. The recovery mechanism enables the optimizer to make speculative optimizations without requiring recovery code.We demonstrate that a rePLay configuration performing ...

16 Sifting out the mud: low level C++ code reuse
Bjorn De Sutter, Bruno De Bus, Koen De Bosschere
November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 37 Issue 11
Full text available: pdf(1.35 MB) Additional Information: full citation, abstract, references, citings, index terms

More and more computers are being incorporated in devices where the available amount of memory is limited. This contrasts with the increasing need for additional functionality and the need for rapid application development. While object-oriented programming languages, providing mechanisms such as inheritance and templates, allow fast development of complex applications, they have a detrimental effect on program size. This paper introduces new techniques to reuse the code of whole procedures at t ...

Keywords: code compaction, code size reduction

17 A region-based compilation technique for a Java just-in-time compiler
Toshio Suganuma, Toshiaki Yasue, Toshio Nakatani
May 2003 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation**, Volume 38 Issue 5
Full text available: pdf(158.62 KB) Additional Information: full citation, abstract, references, citings, index terms

Method inlining and data flow analysis are two major optimization components for effective program transformations, however they often suffer from the existence of rarely or never executed code contained in the target method. One major problem lies in the assumption that the compilation unit is partitioned at method boundaries. This paper describes the design and implementation of a region-based compilation technique in our dynamic compilation system, in which the compiled regions are selected a ...

Keywords: dynamic compilers, on-stack replacement, partial inlining, region-based compilation

18 Compilation and run-time systems: Vacuum packing: extracting hardware-detected program phases for post-link optimization
Ronald D. Barnes, Erik M. Nystrom, Matthew C. Merten, Wen-mei W. Hwu
November 2002 **Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(1.26 MB) Publisher Site   Additional Information: full citation, abstract, references, index terms

This paper presents Vacuum Packing, a new approach to profile-based program optimization. Instead of using traditional aggregate or summarized execution profile weights, this approach uses a transparent hardware profiler to automatically detect execution phases and record branch profile information for each new phase. The code extraction algorithm then produces code packages that are specially formed for their corresponding phases. The algorithm compensates for the incomplete and often incoheren ...

19 Optimising hot paths in a dynamic binary translator
David Ung, Cristina Cifuentes
March 2001 **ACM SIGARCH Computer Architecture News**, Volume 29 Issue 1
Full text available: pdf(890.10 KB)   Additional Information: full citation, abstract, index terms

In dynamic binary translation, code is translated "on the fly" at run-time, while the user perceives ordinary execution of the program on the target machine. Code fragments that are frequently executed follow the same sequence of flow control over a period of time. These fragments form a hot path and are optimised to improve the overall performance of the program.Multiple hot paths may also exist in programs. A program may choose to execute in one hot path for some time, but later switch to anot ...

Keywords: binary translation, dynamic compilation, dynamic execution, run-time profiling

20 Compilation and run-time systems: DELI: a new run-time control point
Giuseppe Desoli, Nikolay Mateev, Evelyn Duesterwald, Paolo Faraboschi, Joseph A. Fisher
November 2002 **Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture**
Full text available: pdf(1.27 MB) Publisher Site   Additional Information: full citation, abstract, references, citings, index terms

The Dynamic Execution Layer Interface (DELI) offers the following unique capability: it provides fine-grain control over the execution of programs, by allowing its clients to observe and optionally manipulate every single instruction---at run time---just before it runs. DELI accomplishes this by opening up an interface to the layer between the execution of software and hardware. To avoid the slowdown, DELI caches a private copy of the executed code and always runs out of its own private cache.In ...

Results 1 - 20 of 39   Result page: **1** 2 next

IEEE Xplore® RELEASE 1.7

Welcome
United States Patent and Trademark Office

IEEE Xp
1 Million D
1 Million U

» **Search Re**

Your search matched **13** of **1043368** documents.
A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance** in **Descending** order.

Refine This Search:
You may refine your search by editing the current search expression or entering a new one in the text box.

binary translation<and>dynamic   Search

☐ Check to search within this result set

Results Key:
JNL = Journal or Magazine **CNF** = Conference **STD** = Standard

1 **Dynamic and transparent binary translation**
Gschwind, M.; Altman, E.R.; Sathaye, S.; Ledak, P.; Appenzeller, D.;
Computer , Volume: 33 , Issue: 3 , March 2000
Pages:54 - 59

[Abstract] [PDF Full-Text (293 KB)] **IEEE JNL**

2 **Dynamic binary translation for accumulator-oriented architectures**
Ho-Seop Kim; Smith, J.E.;
Code Generation and Optimization, 2003. CGO 2003. International Symposium on , 23-26 March 2003
Pages:25 - 35

[Abstract] [PDF Full-Text (362 KB)] **IEEE CNF**

3 **Binary translation: static, dynamic, retargetable?**
Cifuentes, C.; Malhotra, V.;
Software Maintenance 1996, Proceedings., International Conference on , 4-8 Nov. 1996
Pages:340 - 349

[Abstract] [PDF Full-Text (800 KB)] **IEEE CNF**

4 **Dynamic binary translation and optimization**
Ebcioglu, K.; Altman, E.; Gschwind, M.; Sathaye, S.;
Computers, IEEE Transactions on , Volume: 50 , Issue: 6 , June 2001
Pages:529 - 548

[Abstract] [PDF Full-Text (6164 KB)] **IEEE JNL**

5 **Optimizations and oracle parallelism with dynamic translation**
Ebcioglu, K.; Altman, E.R.; Sathaye, S.; Gschwind, M.;
Microarchitecture, 1999. MICRO-32. Proceedings. 32nd Annual International Symposium on , 16-18 Nov. 1999
Pages:284 - 295

[Abstract] [PDF Full-Text (100 KB)] **IEEE CNF**

---

6 **An eight-issue tree-VLIW processor for dynamic binary translation**
Ebcioglu, K.; Fritts, J.; Kosonocky, S.; Gschwind, M.; Altman, E.; Kailas, K.; Bright, T.;
Computer Design: VLSI in Computers and Processors, 1998. ICCD '98. Proceedings., International Conference on , 5-7 Oct. 1998
Pages:488 - 495

[Abstract] [PDF Full-Text (108 KB)] **IEEE CNF**

---

7 **Hardware support for control transfers in code caches**
Kim, H.-S.; Smith, J.E.;
Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on , 3-5 Dec. 2003
Pages:253 - 264

[Abstract] [PDF Full-Text (440 KB)] **IEEE CNF**

---

8 **Dynamic re-engineering of binary code with run-time feedbacks**
Ung, D.; Cifuentes, C.;
Reverse Engineering, 2000. Proceedings. Seventh Working Conference on , 23-25 Nov. 2000
Pages:2 - 10

[Abstract] [PDF Full-Text (656 KB)] **IEEE CNF**

---

9 **Using dynamic binary translation to fuse dependent instructions**
Shiliang Hu; Smith, J.E.;
Code Generation and Optimization, 2004. CGO 2004. International Symposium on , 20-24 March 2004
Pages:213 - 224

[Abstract] [PDF Full-Text (367 KB)] **IEEE CNF**

---

10 **Register liveness analysis for optimizing dynamic binary translation**
Probst, M.; Krall, A.; Scholz, B.;
Reverse Engineering, 2002. Proceedings. Ninth Working Conference on , 29 Oct.-1 Nov. 2002
Pages:35 - 44

[Abstract] [PDF Full-Text (349 KB)] **IEEE CNF**

---

11 **Advances and future challenges in binary translation and optimization**
Altman, E.R.; Ebcioglu, K.; Gschwind, M.; Sathaye, S.;
Proceedings of the IEEE , Volume: 89 , Issue: 11 , Nov. 2001
Pages:1710 - 1722

[Abstract] [PDF Full-Text (232 KB)] **IEEE JNL**

12 **The Transmeta Code Morphing/spl trade/ Software: using speculation, recovery, and adaptive retranslation to address real-life challenges**
Dehnert, J.C.; Grant, B.K.; Banning, J.P.; Johnson, R.; Kistler, T.; Klaiber, A.; Mattson, J.;
Code Generation and Optimization, 2003. CGO 2003. International Symposium on , 23-26 March 2003
Pages:15 - 24

[Abstract] [PDF Full-Text (391 KB)] **IEEE CNF**

13 **Analysis of virtual method invocation for binary translation**
Troger, J.; Cifuentes, C.;
Reverse Engineering, 2002. Proceedings. Ninth Working Conference on , 29 Oct.-1 Nov. 2002
Pages:65 - 74

[Abstract] [PDF Full-Text (303 KB)] **IEEE CNF**